웹기반 분산워크플로우 관리시스템의 프로세스 엔진 설계

이 헌[†] · 박규석[††]

요 약

최근의 웹 환경 확산으로 웹을 기반으로 하는 워크플로우 시스템의 구축경향이 높다. 워크플로우 시스템에서 중요한 기능은 미리 정의된 규칙, 절차 및 조건에 따라 작업흐름을 자동화하는 것으로 정보흐름에 있어서 수동적인 작업을 배제하고 GUI 방식을 지원하며 정보처리의 이동성이 고려되어야 한다. 특히 분산된 업무조직의 작업흐름을 위한 분산워크플로우 시스템은 웹 환경에 있어서 프로세스 로직에 따른 태스크간의 이동제어와 결함허용 및 실시간적 처리가 가능해야 한다. 그러나 기존의 메일 시스템은 전송위주로서 작업흐름을 자동화하려는 경우에 적합하지 못하다. 이러한 문제를 해결하기 위하여 작업흐름 설계가 가능하고 작업흐름 제어를 자동화할 수 있는 분산워크플로우 엔진이 필요하다.

본 논문에서는 웹 메일을 이용하여 작업흐름을 그래픽 인터페이스로 작성하고 이를 절차적 수행이 가능한 스크립트 코드를 생성하는 브라우저를 작성하였으며, 분산환경에서 태스크 처리를 위한 프로세스 스케쥴러 및 서버노드의 결함 발생시 이를 허용하는 전처리기를 지닌 프로세스 엔진을 설계하였다. 이 시스템은 서버의 결함이나 사용자의 부재시 처리를 대행하므로 시스템의 신뢰성과 가용성을 향상시키고, 데이터베이스 기반 워크플로우 시스템보다 비용을 감소시킨다.

The Design of Process-Engine for Distributed Workflow Management System based on Web

Heon Lee[†] and Kyoo Seok Park[††]

ABSTRACT

It has a tendency to build Workflow Systems based on the web with the spread of web environment. The important function in Workflow Systems are to automatize job flow according to the predefined regulations, procedures or conditions. Hence, there needs to consider excluding passive jobs, supporting GUI and a migratory information processing for information flow. Especially, Distributed Workflow System for workflow of a distributed service system should perform transfer control and Fault-Tolerance between tasks based on process logic, and real time processing. However, the existing mail systems being used are just for transmission and it doesn't fit to automatize job flow. To solve the matter, there needs a Distributed Workflow Engine to design workflow and automatize its control.

In this paper, we design a web browser with graphic interface using web mail, a browser creating a script code for a procedural performance. Also, we design a Process-engine has a Preprocessor which tolerates process scheduler for task transaction or server node's faults on distributed environment. The proposed system enhances the reliability and usability of a system and reduces the cost rather than a workflow system based on database, for they execute as proxy for a server's fault or users' absence.

[†] 정회원, 거제대학 전자계산과 부교수, 산학처장
[††] 종신회원, 경남대학교 정보통신연구소 소장

1. 서 론

WfMS(Workflow Management System)의 관리 영역은 조직과 조직이 수행하는 업무, business 프로세스의 개선, 추적, 업무제어, 작업자간의 협업 도모, 작업의 자동화, 정보와 문서의 전자화 및 일관적인 접근과 제어 등으로 요약할 수 있다. 적용된 분야는 보험회사의 사고처리업무, 금융회사의 융자업무, 프로젝트 진행업무, 시스템 관리와 예외처리, 신제품 개발 업무 및 병원연계 가정 건강관리 등이며 관련 연구가 활발히 진행 중이다.

특히, 최근의 웹 환경에 워크플로우 시스템을 적용하기 위한 연구는 수행 프로세스가 공통으로 처리될 수 있거나 복수의 서버에서 수행 가능한 경우 이들의 서비스 시스템을 신속히 선정하는 기능이 미흡할 뿐 아니라 서버의 장애나 사용자 부재인 경우 프로세스 지연 메시지를 전송하는 것으로 방치되므로 업무수행시간이 지연될 수 있고, 사용자 편의를 위한 GUI 방식 처리의 곤란 등 어려움이 있다. 또한 웹 환경은 상대적으로 성능과 신뢰성이 빈약하여 대화형 웹 서비스의 중단으로 인한 서비스 생산자와 소비자에게 치명적인 손실을 줄 수 있어 항상 서비스가 가능하도록 설계되어야 한다. 따라서 일정 수준의 신뢰성과 가용성을 제공하기 위한 결함허용 컴퓨팅 시스템 구축이 필요하다[1-3]. 최근에는 개인용컴퓨터나 워크스테이션 급을 이용한 클러스터링 기법을 통해 시스템을 재 구축하여 하드웨어를 중복시키지 않는 방식이 연구되고 있으나 워크플로우 시스템에 적용하는 문제가 남겨져 있다.

워크플로우 시스템의 정보 전송방법으로는 데이터베이스를 사용하는 방법과 기존의 메일 시스템을 활용하는 방법이 있다. 데이터베이스 시스템은 멀티미디어 정보를 다수 사용자에게 전달할 경우 많은 공간이 요구되므로 지속적인 관리가 어렵고, 메일 시스템은 개별적인 정보 관리로 데이터베이스 접근 비율을 감소시켜 비용의 분산을 가능하게 하지만 시스템이 안정적이지 못하다.

본 논문에서는 이러한 특성을 고려하여 기존의 메일 시스템을 활용한 작업흐름 자동화를 설계하고 웹 환경에 적합한 워크플로우 시스템을 작성하였다. 제안하는 웹기반 분산워크플로우 시스템은 작업흐름을 프로세스 별로 나누어 태스크를 산출하고 아이콘화 하여 절차적으로 수행하는 프로세스 엔진과, 서버 노드의 결함허용을 지원하는 전처리기, 태스크의 서비스 서버를 선정하고 보증여부의 판정을 위한 태스크 스케줄러 등으로 구성되며, 본 시스템은 전체 시스템의 가용성과 신뢰성을 향상시킨다.

2. 관련 연구

2.1 워크플로우 시스템

워크플로우는 미리 정의된 일정한 규칙과 절차에 따라, 한 사람에서 다른 사람에게 전달되는 과제, 문서 및 정보의 흐름을 자동화하는 시스템으로 전체 또는 부분적으로 비즈니스 프로세스를 컴퓨터화하여 편리하게 하거나 자동화한 것이다.

워크플로우 관리시스템은 업무 흐름의 자동화와 정보 및 문서 전달의 전자화, 그리고 일관적인 데이터 접근과 제어를 통해 업무 프로세스의 개선, 통제, 관리 및 공동작업을 지원하는 소프트웨어로서 문서보다는 과정을 중시한다[4,5].

워크플로우 관리시스템은 컴퓨터 표현에 의해 정해진 실행 순서에 따라 워크플로우 로직을 실행한다. 정보의 흐름과 관련된 기업 또는 부서 및 관련자들간의 업무 연계를 돕고, 정보의 유연한 전달을 위한 기능을 제공하며, 다양한 작업활동의 순서와 관련된 사람간의 정보기술자원을 공유함으로써 업무 프로세스의 자동화를 제공하는 시스템이다[5-7].

워크플로우 시스템은 초창기의 SCOOP[8], Office-Talk[9] 등이 개발된 이후로 많은 개발 회사의 프로젝트들로부터 발표가 있었다. 이와 관련한 표준화로는 WfMC(Workflow Management Coalition)의 5개 인터페이스 및 메타모델이 있고[3], PIF(Process Interchange Format) Work Group은 프로세스 정보 교환 형식에 대한 표준을 연구하고 있으며[10], NIST(National Institute of Standards and Technology)는 프로세스 표현언어에 대한 필요사항을 명세하였다[11].

WfMC에서 표준으로 제시한 일반적인 워크플로우 시스템의 구조는 그림 1과 같으며, 본 논문에서 참조 모델로 한다.

WfMC의 구성요소는 워크플로우 시스템에서 다양한 기능을 지원하는 소프트웨어 요소, 하나 또는 하나 이상의 소프트웨어에 의해 사용되는 시스템 정
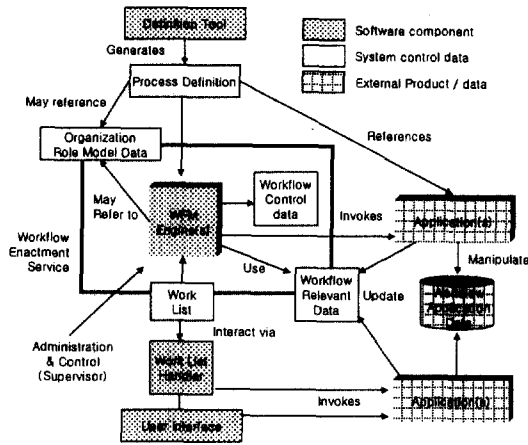
그림 1. 일반적인 워크플로우 시스템 모델

의와 제어 정보 그리고 워크플로우 시스템에서 직접 작성하지 않고 참조하는 정보 등이다. 또 구조는 프로세스를 정의하는 정의 도구, 워크플로우를 처리하는 워크플로우 엔진, 데이터 저장 시스템 및 사용자 인터페이스와 사용자 응용프로그램 등으로 구성된다.

워크플로우 관리시스템에 대하여 프로세스 및 관련 업무 데이터, 태스크간의 전이를 나타내는 트랜지션 데이터, 프로세스가 사용하는 워크플로우 관련 데이터, 응용 프로그램 관련 데이터 등 여러 정보를 기존의 파일이나 데이터베이스 시스템을 개선시켜 분산환경에서 워크플로우와 연동하여 운영 가능한 정보저장소에 대한 연구도 있다[12]. 여기서 정보저장소는 복잡한 정보처리 환경에서 보다 효율적인 정보자원의 관리를 위해 각 응용 시스템들이 요구하거나 사용하고 있는 모든 정보자원을 통합, 저장 및 관리할 수 있는 시스템을 의미한다[13]. 최근에는 웹브라우저의 확산으로 웹 기반 워크플로우 시스템 개발과 사용자 편의를 위한 GUI방식의 인터페이스를 위한 자바기반 또는 전용 어플리케이션의 개발이 활발하다.

2.2 분산실시간 시스템

분산시스템은 메모리의 공유없이 각 프로세서들이 통신망(communication network)을 이용하여 메시지를 교환하는 방법으로 상호 결합되는 컴퓨팅 시스템이며, 프로세서 자원 중복성(multiplicity)은 단일 프로세서 시스템에 비해 보다 높은 가용성(availability)을 제공한다[14]. 사용자에게는 지역적으로 분리되어 있는 여러 장소의 장비와 자원을 하나의 통합된 시스템에서 처리하는 듯한 투명성을 제공하므로 인터넷의 이용 증가와 더불어 구축환경의 새로운 전환기를 맞고 있다.

실시간 시스템은 엄격한 시간제약(deadline)과 높은 신뢰도를 만족시키며 상호 작용하여 제어를 수행한다. 따라서 시스템 측면에서 실시간 프로세스는 보다 체계적으로 통제되어야 하며[15], 다양한 기능을 동시에 처리하기 위해 응용 프로그램을 여러 개의 모듈로 나누어 각 모듈들을 독립적인 태스크로 구동시키는 방법이 요구된다.

분산실시간 시스템은 지역적으로 분산되어 있거나 국부적인 데이터 및 사건을 시스템으로 받아 들여 주어진 시간이내에 처리한 후 되돌려 보내는 시스템이다. 실시간 운영체제의 다중 작업은 시스템의 자원들을 효과적으로 이용하기 위해 각 태스크를 제어하며, 태스크들의 이식성 개념에 부합되도록 하드웨어와는 종속적인 부분과 독립적인 부분으로 구분하여 스케줄러와 사건(event)에 따라 상호 연결한다.

2.3 분산시스템 환경의 워크플로우 시스템

워크플로우는 작업흐름을 통제 관리하는 통제객체를 비롯하여 업무를 수행하는 다수의 담당자 또는 컴퓨터 수행객체 및 공유자원이 네트워크를 통하여 연결되어 지는 구조이다. 각각의 컴퓨터들은 분산객체환경에서 구성요소로써 인식될 수 있으며, 구성요소들은 서비스를 제공할 수 있는 전체 자원을 의미한다. 실재 업무환경에서 자원은 인간, 컴퓨터 및 특정 목적을 위한 프로그램과 하드웨어 장비를 포함하며, 각 구성요소들은 고유의 기능을 갖는 독립적인 수행객체이다. 따라서 업무처리흐름을 나타내는 워크플로우는 네트워크를 통하여 연결된 객체들이 공유자원과 각 객체가 제공하는 서비스를 이용하는 분산객체환경을 기반으로 모델링된다.

WfMC에서 워크플로우 제정 서비스는 중앙집중방식이나 분산방식으로 구현되며, 하나 이상의 워크플로우 엔진으로 구성된다.

본 논문에서는 분산 실시간 시스템 환경에서 서버노드마다 하나의 프로세스 엔진을 지니고 프로세스 정의에 따라 엔진들이 상호 협력하여 태스크 실행을 나눌 수 있도록 하였다.

2.4 결함허용 시스템

결함허용을 위한 연구로는 웹 서비스와 연계하여 결함발생시 웹 서버의 부하 분배기에 의한 클러스터링 시스템을 구축하고, 클러스터 내에 있는 특정 서버에 부하를 집중시키지 않도록 하는 부하 분산방법이 있다. 부하 분배 스케줄링 방법으로는 라운드 로빈 스케줄링, 가중 라운드 로빈 스케줄링, 연결된 사용자 요구 수가 제일 적은 곳을 우선적으로 할당하는 최소 연결 스케줄링 및 실시간 환경에 적합한 데드라인 우선 정책 등이 있다.

SWEB은 워크스테이션들의 네트워크와 분산 메모리를 사용하는 웹 서버로서 모든 서버가 전체문서를 포함하지 않으며 LAN을 통해 다른 서버노드의 문서를 가져오는 대표적인 프로젝트의 하나이다 [16]. 이 때 가장 적은 소요시간이 예측되는 서버노드를 선택하기 위한 기준으로 CPU 처리시간, 디스크 대역폭 및 네트워크 지연 등을 고려하여 최소인 것을 선택한다.

또한 각 서버노드의 평균 유휴시간(idle time)을 이용하여 부하를 분배시키는 방법이 있으며, CPU 부하나 큐 길이 등을 이용한 부하 분산 기법, 사용자의 요구에 대해 미리 정의된 문서의 접근 확률에 따라 서비스 될 서버노드를 결정하여 하나 이상의 서버에서 장애가 발생되었을 경우 네트워크 플로우 기반 알고리즘으로 분배 확률을 재 계산함으로써 서버노드의 결함을 허용하는 방법도 있다[17].이 방식은 정적처리 방식으로 문서 크기가 유사하지 않거나 전송시간이 다를 경우에 부하불균형이 나타날 수 있다.

한편, 각 서버노드의 CPU 부하 정보나 idle time을 이용하는 부하 분배의 경우, 클러스터링 시스템의 부하 분배기는 모든 서버노드들을 주기적으로 모니터링 하게 되고, 부하 모니터링 주기에 대한 결정과 이에 따르는 오버헤드를 최소화하는 것이 중요하다[13].

본 논문의 프로세스 스케줄링 알고리즘은 이러한 문제점들을 최소화하기 위해 나머지 서버노드들로 시스템을 재구성하고, 태스크 처리시간을 중심으로 부하를 최소화하는 정책으로 분배하므로 태스크를 신속하게 수행할 수 있도록 하며, 워크플로우 엔진으로서의 기능을 갖게 한다. 또한 워크플로우 시스템을 메일시스템 기반으로 트랜잭션할 수 있도록 구성하여 DB 기반 워크플로우가 지닌 소요공간의 과다 사용을 줄이고 전체 오버헤드를 감소시킨다.

3. 시스템 모델

분산워크플로우 시스템 모델은 그림 2와 같다.
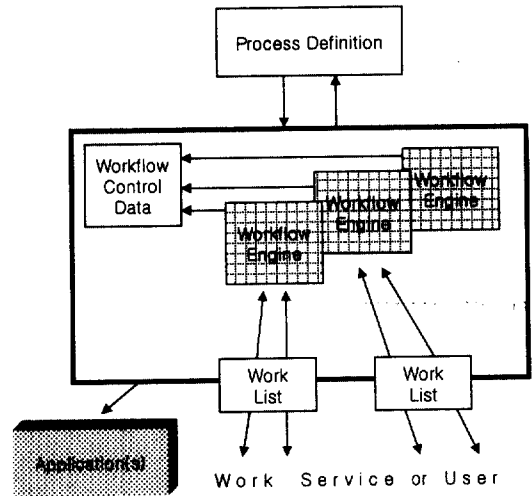
그림 2. 분산 워크플로우 시스템 모델

그림 2에서의 워크플로우 콘트롤 데이터는 다수개의 워크플로우 엔진을 통제하기 위하여 해당 정보를 관리하는 곳으로 전체 분산 워크플로우 시스템의 주요 정보이다.
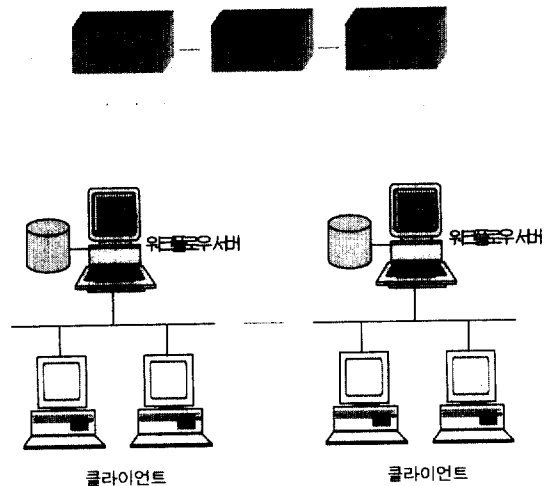
그림 3. 분산워크플로우 구성 모델

그림 3에서의 분산 워크플로우 구성모델은 각각의 노드마다 워크플로우 서버를 지닌 독자적인 시스

템의 결합으로 구성되며 해당되는 업무를 나누어 수행한다.

분산워크플로우 시스템 모델의 작업흐름에 대한 시나리오는 호스트기반 모델, 공유 파일 모델, 전자메일 모델, 메시지 전송 모델 및 프로시듀어 호출 모델 등이 있으며, 본 논문에서는 메일 시스템 모델을 이용한다.

그림 4는 워크리스트 처리가 클라이언트/서버 시스템에서 수행되는 과정을 나타낸다.
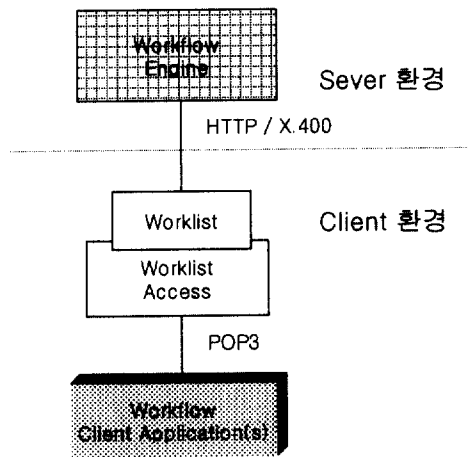
그림 4. Worklist 처리 개요

3.1 처리 모델

메일 시스템은 클라이언트와 서버 영역으로 나누며, 정보전달 및 수행 모델은 그림 5와 같이 나타낼 수 있다. 이것의 기능을 확장하여 분산실시간 시스템에 적용할 수 있도록 하는 경우 각 서버노드마다 자신이 지니고 있는 DB를 이용하고 상호간에는 메일을 이용한 메시지를 통해 작업흐름을 제어할 수 있다.

메일 브라우저는 정보전달 및 상호 동작을 위하여 SMTP 프로토콜을 이용하고, 전송된 정보를 획득하기 위하여 POP3 프로토콜을 지원한다. 특히 작업의 흐름을 태스크 단위로 그래픽 프로세스 모델링 방식을 이용하므로 정확성과 효율성을 향상시키고, 작성된 프로세스 흐름을 나타내는 방향성 그래프는 스크립트 언어 코드로 변환되므로 사용자가 직접 작성하는 프로그램은 없다.

여기서 작업흐름을 나타내는 코드는 수행을 위하여 HTTP 프로토콜을 이용하여 프로세스 엔진을 가
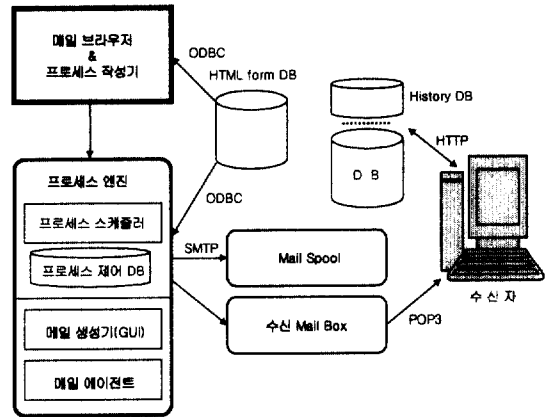
그림 5. 워크플로우 수행모델

지고 있는 웹 서버에 등록된다.

프로세스 엔진은 프로세스 제어를 위한 DB와 프로세스 스케줄러로 구성되고, 데이터베이스 구동방식과 메일 구동방식을 혼용한다. 프로세스 스케줄러는 해당 태스크를 구분하여 지역 태스크는 작업수행 서버노드로 전송하고 전역 태스크는 제어 DB의 정보를 이용하여 수행할 서버를 선정하기 위한 태스크 스케줄링을 한다. 프로세스 제어 DB는 프로세스 스케줄링과 태스크 스케줄링에 필요한 정보를 제공하고 수행상태를 모니터할 수 있게 하며, 각 서버노드의 상태를 파악하고 있다.

프로세스 스케줄러는 웹 및 분산실시간 시스템에서 구현될 수 있으며, 시스템 구조는 워크플로우 서버를 중심으로 다수의 클라이언트들이 연결되고, 워크플로우 서버는 자신이 필요로 하는 DB를 분산하여 지닌다. 이들 각각은 작업 프로세스를 발생시킬 수 있으며 동시에 프로세스 실행 노드가 될 수 있다. 그리고 결합허용과 전역 태스크의 실행 보증을 향상시키기 위하여 전처리기를 둔다.

3.2 메일 브라우저

제안하는 메일 브라우저는 기존의 유닉스 시스템의 MTA(Mail Transfer Agent)와 클라이언트 UA(User Agent) 방식을 이용하고, MIME 형식을 지원함과 동시에 HTML 형식의 문서를 지원하여 미디어 자원을 인라인 방식과 분산시스템을 위한 접근이 가능하도록 하였으며, 프로세스와 문서의 흐름을 제어할 수 있는 그래픽 프로세스 모델링 도구와 스크

립트 생성기를 지닌다. 메일 브라우저는 크게 송신부와 수신부 및 프로세스 작성기로 구성되며, 세부 구성으로는 네트워크 모듈, 코드생성 모듈, UA 모듈, 사용자 인터페이스 모듈 및 Viewer 모듈 등이 있다.

사용자 인터페이스 모듈은 메일 브라우저를 사용할 수 있도록 GUI 환경을 제공하며, 메일 부분의 UA 수행을 병행하고 메시지를 받아 디스플레이 시켜준다. 네트워크 모듈은 프로토콜 매니저로서 메일을 전송하는 SMTP 프로토콜과 메일 수신을 위한 POP3 프로토콜 및 분산 자원을 받기 위한 HTTP 프로토콜로 구성된다. 메일박스 매니저 모듈은 메일서버에서 전송 받은 메일들을 지역 시스템에 저장시키고 관리하며 파일 매니저 형식을 따르고, 컨버터 매니저 모듈은 전송받은 메시지를 텍스트 형식으로 변환하거나 디스플레이 구조로 변환시키는 기능을 지니며, HTML과 MIME를 텍스트로 변환시키는 알고리즘을 지닌다.

코드생성 모듈은 스크립트 생성기를 통해 프로세스 작성기의 기능을 부분적으로 담당하며, 그래픽 프로세스 모델링 도구에서 작성된 작업 흐름의 모델에 따라 스크립트 언어를 번역해 준다. 그래픽 아이콘 생성기는 그래픽 프로세스 모델링 도구에 사용되는 사용자 인터페이스의 집합과 규칙에 따라 인터페이스의 적용범위를 설정할 수 있는 기능으로 구성되며, Viewer 매니저 모듈은 인라인 메타 데이터와 분산된 메타 데이터를 디스플레이 하는 기능으로 MIME 헤더 형식에 따라 표현된다.

1) 송신처리 구조

메일 브라우저는 데이터 흐름 스케쥴 정보전송과 수신된 문서의 응답 등 두 측면에서 작용한다. 데이터 흐름 스케쥴 정보는 프로세스 엔진에 스케쥴 정보를 등록하기 위하여 HTTP 프로토콜을 이용하여 전송되며, 이 스케쥴 정보에 따라 프로세스 엔진의 프로세스 스케쥴러가 판정한 메시지 정보는 SMTP 프로토콜을 이용하여 메일 박스로 전달된다. 메일 브라우저의 송신처리 과정은 그림 6과 같다.

메시지의 전송 및 응답 정보는 전자메일 표준 프로토콜인 SMTP를 이용하고, MTA 상호 간의 NVT ASCII를 이용하여 메시지를 교환하며 클라이언트에서 서버로 명령을 전송한다. SMTP 명령은 HELO, MAIL, RCPT, DATA 및 QUIT를 사용하여 수행된다.
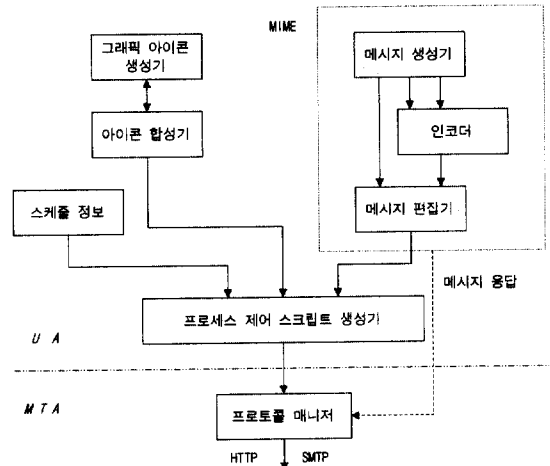
그림 6. 송신처리 과정

데이터 흐름 스케쥴 정보는 그래픽 아이콘 생성기에 의해 문서의 흐름을 작성하게 되며, 작성된 스케쥴 정보는 MIME 처리된 메일 문서와 함께 프로세스 제어 스크립트 생성기를 이용하여 스크립트 언어를 생성한다.

2) 수신처리 구조

프로세스 엔진에서는 데이터 흐름 스케쥴 정보 스크립트를 구문 분석기로 메시지와 스케쥴 정보를 추출하게 되고, 프로세스 스케쥴러는 스케쥴 정보를 이용하여 메시지 부분을 라우팅할 수 있도록 한다. 라우팅 과정에서 사용자 메일 박스에 메시지 부분이 전달되며 메일 브라우저에 의하여 수신된다. 메일 브라우저의 수신처리 과정은 그림 7과 같다.
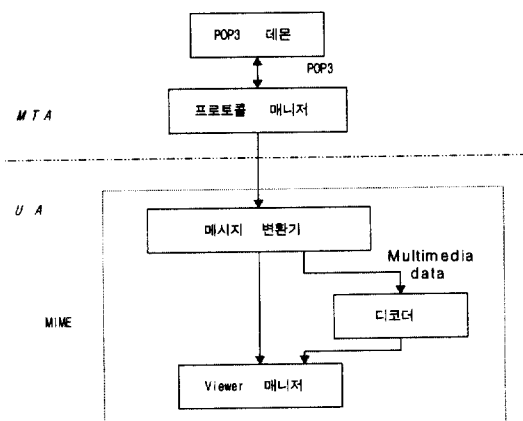
그림 7. 수신처리 과정

전송된 전자메일 문서를 디스플레이하기 위하여 문서 분석기가 있고 분석된 데이터는 실제 미디어 자원을 표현하기 위해 디코딩 된다.

3.3 그래픽 프로세스 구조

메일 브라우저는 문서의 흐름을 제어할 수 있는 그래픽 프로세스 모델링 도구를 지녀야 한다. 기존의 작업흐름 설계 도구는 작업흐름을 나타내는 다이어그램을 이용해 표현하지만, 본 논문에서는 태스크의 개념적 의미관계로 설정하는 방법을 제안한다.

제안 방법에서 그래픽 프로세스 모델링 구조는 작업흐름의 순서를 설정할 수 있으며, 문서의 흐름관계를 아이콘들간의 연결로 표현하고 각각의 아이콘은 태스크 속성을 지니는 방향성 그래프를 구성시킨다.

태스크의 수행속성은 행위자, 역할, 수행조건, 트리거링 타입 등으로 구성되어 작업흐름관계를 표현한다. 이 때 프로세스에는 수신측에서 태스크를 추가하거나 서브 태스크를 만들어 변경할 수 있으며 단순 참조는 참여측에서 언제든지 가능하나 수정 할 수는 없다.

이러한 그래픽 인터페이스는 그래픽 편집 기능 및 관련 아이콘을 제공하고, 개체간의 흐름관계를 나타내기 위해 각각의 개체에 태스크 개념을 제공한다. 또한 태스크 행위들 간의 데이터 또는 제어 흐름들이 표현될 수 있도록 조직을 모델링 하며, 태스크의 관련 데이터를 나타내고 작업흐름 관리를 위한 스크립트나 해당 정보를 생성시킨다.

본 논문에서는 그래픽 인터페이스를 위해 데이터 흐름제어 시스템의 그래픽 프로세스 모델링 도구인 OMT의 객체 모형화 기법을 변경, 추가하여 이용한다.

기본적인 심벌로는 프로세스 전 과정에 시작, 처리, 종료 태스크가 있고 응답요구, 처리 진행 및 AND와 OR 조건을 나타내는 논리 흐름, 피드백과 루틴을 나타내는 심벌과 전역수행 태스크를 나타내는 확장 심벌을 사용한다.

그림 8. 확장 그래픽 심벌

3.4 프로세스 엔진

워크플로우는 프로세스와 자원 그리고 프로세스 제어로 구성된다. 하나의 프로세스는 다수의 단위 업무인 태스크로 구성되며, 각 프로세스는 실행상태에서 다수의 인스턴스를, 태스크는 태스크 인스턴스를 생성한다. 자원은 인력과 동일한 역할을 수행하는 일련의 인력 집단인 역할집단(Role)으로 나누어지고, 프로세스 제어는 프로세스 인스턴스와 태스크 인스턴스의 수행제어를 담당한다.

이를 위해 워크플로우에 포함되는 정보의 형태는 프로세스 정의 정보, 조직체와 규칙 정보, 프로세스 제어 데이터, 워크플로우 관련 데이터, 작업 리스트(Work List) 등이 있다.

제안하는 프로세스 엔진은 모든 절차나 워크리스트 내의 단계들 그리고 각 단계별 규칙 등을 알고 있는 작업흐름 프로그램 내의 구성요소로서의 프로세스가 다음 단계로 옮길 준비가 되었는지를 판정한다. 이를 위하여 프로세스를 해석하고 제어하며 활동 순서를 정의하고, 정의된 활동들은 사용자 업무 요소(work item)에 추가 시켜 필요한 응용 프로그램을 가동시켜준다. 이 때 프로세스 엔진의 역할은 업무 정의 해석, 프로세스 인스턴스 생성 및 시작, 종결, 일시정지, 재시작 등의 실행관리, 태스크 종결시의 프로세스 상태 수정, 태스크서비스에게 업무 전달, 프로세스 진행 과정상의 항목을 생성하고 태스크간의 이동제어, 태스크나 프로세스가 데드라인을 넘겼을 때의 조치 및 진행 중인 프로세스에 대한 감독과 관리 등이다.

프로세스 엔진은 프로세스 제어를 위해 전처리기와 태스크 스케줄러로 구성된 프로세스 스케줄러를 지니며 스케줄 정보를 받아들이기 위해 웹 서버 및 데이터베이스와 연계된다. 프로세스 엔진의 구성도는 그림 9와 같다.

WCS(Workflow Control Script) 처리기는 수행코드 데이터베이스에서 프로세스 흐름 제어 스크립트 코드의 어휘 및 구문분석을 수행하고, 태스크 스케줄 정보 테이블과 태스크 수행 상태 테이블을 생성하며 스케줄 태스크 테이블에 수행코드를 등록한다. 언어
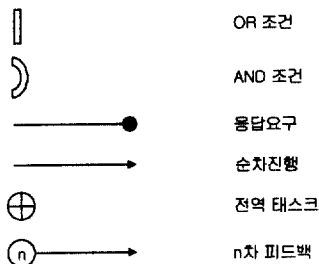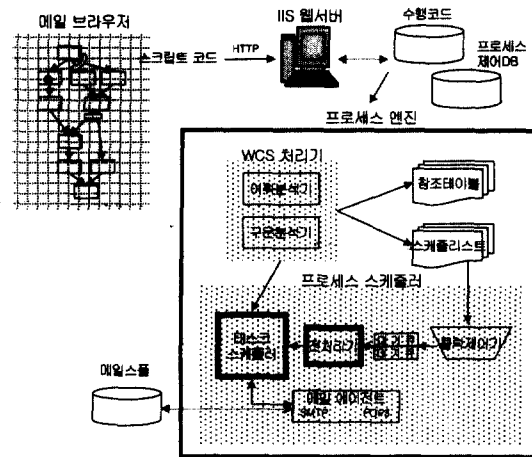
그림 9. 프로세스 엔진 구성도

분석기에서 의미분석을 마치면 흐름 제어 코드마다 태스크 스케줄러의 쓰레드를 할당받아 스케줄 정보로서 관리하며, 생명주기에 의해 상태를 변경한다.

프로세스 스케줄러의 세밀한 구성은 클럭 제어기, 메일 에이전트, 전처리기, 태스크 스케줄러 및 관련 정보를 저장하는 공간 등이다. 정보 저장 공간의 대기 큐는 스케줄 대기 큐와 응답 대기 큐가 있으며, 클럭 제어기는 시간 알람을 이용하여 작업 제어 리스트를 스케줄 대기 큐로 보내고, 대기 큐의 내용은 태스크 스케줄러 또는 전처리기에서 사용한다.

메일 에이전트는 메일 라우팅 기능과 메일 수집기능을 가지며, 라우팅 기능은 메일을 전송하는 SMTP 프로토콜로 구성되고 스케줄링된 리스트 정보를 이용하여 데이터베이스에 저장된 업무나 문서 내용을 각 메일박스로 전송한다. 메일 수집기는 서버가 지정한 일정 시간에 전용 메일 스풀에서 상호 응답이나 대기 지연 메시지를 추출하여 응답 대기 큐에 적재한다.

전처리기는 서버노드에서의 결함을 허용하고 태스크 중 일부에 대한 수행 보증여부를 판단한다. 태스크 스케줄러는 스케줄 대기 큐와 응답 대기 큐의 정보를 이용하여 수행 절차를 보증하며 진행시킨다.

1) WCS(Workflow Control Script) 처리기

메일 브라우저에서 작성된 WCS는 프로세스 엔진에서 수행 처리되기 위해 내부 표현으로 변환하며, 내부 표현은 데이터베이스에 저장되어 시스템 종료나 모니터링 정보를 제공한다. WCS 처리기에 의해 관리되는 테이블은 프로세스 테이블, 프로세스 상태 테이블, 스케줄 리스트 테이블, 스케줄 상태 테이블 등이며, 스케줄 정보로는 작업번호, 스케줄 번호 및 응답 코드 등을 이용하여 데이터 흐름을 제어할 태스크 ID를 할당하고 스케줄 리스트 테이블에 등록한다.

2) 프로세스 스케줄러

프로세스 엔진에 의해 산출된 결과에 따라 프로세스의 매 단계마다 해당 서버를 통해 메일로 업무의 발생, 진행, 결과 등을 알려 주며 POP3를 이용한다. 단, 프로세스는 작업흐름과 업무에 따라 여러 가지 유형을 가질 수 있으며, 본 논문에서의 업무 프로세스 유형은 다음의 두 가지 특징을 갖는다. 서로 다른 프로세스 엔진에 의해 관리되는 단위 작업들이 불규칙적으로 혼합되어 하나의 복합된 프로세스를 이루는 것과, 하나의 단위 작업이 다른 엔진에서 작동되는 여러 하부 업무 프로세스를 생성시키는 것이다. 수행은 프로세스의 태스크를 직접 실행하여야 하는 경우와 단순히 참조하는 경우로 구분할 수 있고, 피드백하는 경우도 포함된다. 또한 태스크는 지정된 서버노드에서 실행되어야 하는 엄격한 지역 태스크와 전체 또는 일부 그룹에서 공통으로 실행될 수 있는 전역 태스크가 있다.

프로세스 스케줄러는 결함허용을 위해 전처리기를 두며, 결함 발생시 지역 태스크의 경우에는 특별히 지명된 대리 사용자가 처리할 수 있거나 지연되고, 전역 태스크는 부하 균형을 고려하여 실행 노드를 신속히 선정해야 한다. 모든 프로세스는 실시간으로 처리를 요구하고 해당 데드라인을 만족시키지 못하면 다시 제출되거나 거부된다.

각 웹 서버노드는 동적으로 프로세스를 발생시키고, 자신의 처리 태스크에 대하여 서브 태스크를 발생시킬 수 있다. 전처리기와 태스크 스케줄러의 관계는 그림 10과 같다.

■ 클럭 제어기

클럭 제어기(Clock Controller)는 WCS 처리기가 생성한 스케줄 리스트에서 지정된 시간에 수행될 스케줄 리스트를 추출하여 스케줄 대기 큐에 저장하며, 태스크 스케줄러 또는 전처리기와 상호 협조적으로 동작한다. 태스크 스케줄러에서 수행된 스케줄에서 데드라인을 가지는 태스크는 데드라인 정보가 스케줄 상태 테이블, 스케줄 리스트 테이블을 이용하여 클럭 제어기의 스케줄 대기 큐에 등록된다.
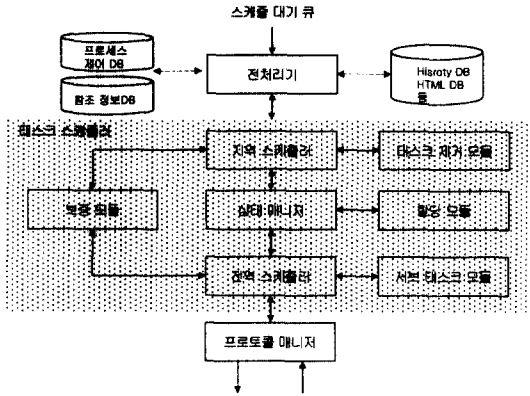
그림 10. 전처리기와 스케줄러 구조

또 태스크 스케줄 정보 중에서 태스크 시작 시간을 추출하여 스케줄 대기 큐에 저장시키며, 메일 에이전트에 의해 응답 대기 큐에 저장된 정보와 함께 태스크 스케줄러에 제공된다.

■ 전처리기

전처리기(Preprocessor)는 도착한 태스크와 그에 대응하는 정보 DB 및 프로세스 제어 DB 등을 이용하여 해당 태스크의 보증을 시도한다. 여기서 태스크는 주기적으로 발생한 것과 비주기적으로 발생한 것을 구분하고, 주기적 태스크는 곧장 수행을 위한 판정을 위해 지역 스케줄러로 보낸다. 단, 서버노드나 웹 상의 결함(fault)이 발생한 경우에는 데드라인과 서비스 시간을 고려하여 긴급하지 않는 경우, 태스크 진행 과정을 History DB에 보관시키고 해당 메시지를 발생시켜 실행을 기다리도록 한다. 실행이 긴급한 것 중에서 공통 태스크는 전체 시스템에서 해당 클러스터 정보를 참조하여 수행 노드로 전송을 준비한다. 이 과정에서는 정밀한 노드 선정이 되지 못하므로 전역 스케줄러에서 보증을 확인 받아야 하며, 그 순간까지 작성된 정보를 메일과 메시지를 통하여 공유할 수 있도록 제공한다.

다른 서버노드로부터 실행 전송된 전역 태스크나 결함으로 인한 대리 수행 요구 태스크는 전처리기가 관리하고 있는 factor를 참조하여 보증을 시도한다. 이 때 전처리기의 정보는 프로세스 엔진의 수행 대기 큐가 변경될 때마다 factor 값을 산출하여 갱신시킨다. 따라서 factor 값에 의한 보증 판정은 엄격하지 못하나 신속성과 오버헤드 측면에서 유리하며, 업무에서 분산워크플로우 시스템이 반복적으로 사용되는 경우 그 History를 통해 경험적 정보 누적으로 엄격성을 높일 수 있다.

결함이 발생한 노드나 웹 서버의 전처리기는 이에 대한 정보를 최소한 공유시킬 수 있도록 결함으로부터 자유로워야 하며, 결함 허용(Fault-Tolerant)을 위해 다음의 정보를 관리한다.

Ppr_id	S_fc	F_ch	Sn_1	....	Sn_n	Sp_n
				....		

그림 11. 전처리기 정보 테이블

Ppr_id 는 전처리기 ID로 웹의 경우 IP 주소와 대응되며, 분산시스템에 있어서는 노드 식별자이다. S_fc는 전처리기 factor값으로 서비스의 여유분을 나타내는 값으로 수행 대기 큐의 변경시 식(1)을 이용하여 수정된다. 결함이 발생한 경우에 유효한 값을 갖게 되는 F_ch는 전역 태스크의 신속한 실행 노드를 선정하는데 사용하며, 자신에게로 전송되어 올 전역 태스크의 오버헤드를 줄이도록 한다. Sn_1~n은 공유(share)노드로써 시스템 구성에 따라 다수 개로 설계할 수 있으며, 작업이 사람에 의해 수행되어야 할 경우에도 유용하다. 즉 사용자에게 메시지나 문서 정보가 전송되어야 할 때, 부재일 경우 대리인 또는 대행 조직으로 적극적 메시지를 전송할 수 있도록 한다. Sp_n 은 중복(spare)노드 정보로서 실행이 가능한 노드가 하나 이상 있는 경우 기록되며 결함허용을 위하여 우선적으로 검토한다. 단, 전처리기가 지니고 있는 정보는 스케줄링을 신속히 하기 위한 최소 정보이며, 상세한 정보는 프로세스 제어 DB에서 참조한다.

■ 태스크 스케줄러

태스크 관리를 위한 정보는 다음과 같다.

T_id	P_id	Pri	J_ty	D_ln	Sv_t	St_t	Ag_n	Pe_c	C_id

그림 12. 태스크 테이블 구조

T_id는 태스크 식별자, P_id는 프로세스 식별자이며, Pri는 태스크의 우선 순위, J_ty는 태스크의 타입을 나타낸다. 그리고 D_ln은 태스크의 데드라인,

Sv_t는 서비스 시간, St_t 는 시작 시간이고 Ag_n은 대리인을, Pe_c는 태스크의 주기성 여부를 나타내며 C_id는 클러스터 식별자로 태스크의 분리 가능여부이다. 이 외에도 태스크 이름, 태스크 기술내용, 문서 이름, Due time, Split 타입, Merge 타입, Role 이름, Abort 가능여부, Reject 가능여부, 제어 가능여부 등이 포함된다.

태스크 타입은 지역 태스크와 전역 태스크로 구별하고, 대리인은 상징적으로 태스크 타입에 따라 대응할 수 있으나 결함 허용에까지 적용할 수 있기 때문에 분리한다. 시작 시간은 참고 정보이며, 서비스 시간은 프로세스 정의에 의해 설정되고 History DB로부터 갱신될 수 있도록 하여 경험적 작업 흐름이 가능하다. 태스크의 종류는 Regular, Dummy, Irregular 및 Agent가 있으며, Split타입은 AND, OR와 조건에 따라 흐름이 변경되는 CONTROL 및 default 값인 NONE을 둔다. Merge 타입은 Split 타입과 동일하게 적용된다.

태스크 보증에 필요한 제어변수와 조건 정보는 다음과 같다.

RT_id	CV_id	CV_nm	CV_dc	CV_vl	CV_ty

그림 13. 제어변수 테이블

제어변수 테이블은 관계된 태스크, 제어변수 식별자, 제어변수명, 제어변수 내용, 제어변수 값 및 타입 등으로 구성된다.

지역 스케줄러는 전처리기로부터 도착한 프로세스에서 해당되는 태스크를 분리하여 실행 보증을 위해 보증 모듈을 호출한다.

보증 모듈은 태스크 테이블을 이용하여 세부적인 실행 여부를 판정하기 위해 데드라인을 참고한다. 워크플로우 시스템에서 발생하는 프로세스는 긴급정도별 데드라인을 지니고 있는 실시간 시스템을 전제로 한다. 따라서 보증 모듈에서는 먼저 태스크의 주기성을 분석하고 태스크가 지니고 있는 데드라인 우선 정책과 surplus를 이용하여 실행 보증여부를 판단한 후, 보증될 경우 보증 수행 대기 태스크들을 고려하여 실행 스케줄을 다시 작성하여 해당 정보를 보관하고 시작시간을 예약시킨다. 만약 프로세스나 태스크에 우선 순위(priority)가 있는 경우 surplus를 늘리기 위해 태스크 제거 모듈을 호출하여 제거 태스크를 정하고 surplus를 재 산출한 후 스케줄링을 행한다.

surplus 산출 방법은 식(1)과 같다.

$$\text{surplus} = Wt - \sum_{i=1}^{n}(Pt_i(Wt/P)) - \sum_{j=1}^{m}(Gt_j) \quad (1)$$

여기서 Wt는 작업가능 시간, Pti는 주기적 태스크의 서비스 시간, P는 주기적 태스크의 주기, Gtj는 주기적 태스크를 제외한 보증 태스크의 서비스 시간이다. n과 m은 각각 Wt 내에서의 주기적 태스크의 개수 및 보증된 비주기적 태스크의 개수를 나타낸다.

태스크 제거 모듈은 보증된 태스크들 중에서 우선순위와 데드라인을 고려하여 제거 태스크를 선정하고 새로이 보증될 태스크의 서비스 시간을 반영시킨다. 여기서 제거된 태스크와 보증 모듈에서 실행이 보증되지 못한 태스크는 프로세스 제어 DB를 참조하여 전역 태스크 여부를 식별하고, 전역 태스크인 경우에는 태스크 제거 모듈을 통해 지역 스케줄러로부터 제거하여 전역 스케줄러로 보낸다. 전역 태스크가 아닌 경우에는 대기 큐에 넣어 다시 스케줄링에 참여될 수 있게 한다.

전역 스케줄러는 지역 스케줄러에서 보증되지 못한 태스크들 중 전역 태스크 또는 결함허용을 위한 대리 서비스(spare) 지정에 의한 태스크 등을 보증한다. 여기서도 보증되지 못한 태스크는 다시 서브 태스크 모듈로 보내어져 다수의 서브 태스크를 구한다. 만약 서브 태스크를 작성할 수 있다면 하나 또는 그 이상의 노드에서 보증될 수 있다.

상태 매니저는 지역 및 전역 스케줄러에서 보증된 태스크의 모든 정보를 관리하고, 서브 태스크를 확인하여 스케줄링 과정에서 참조되게 한다. 또한 보증된 태스크의 작업수행 과정을 진행시킨다.

태스크 제거 모듈은 지역 스케줄러가 도착한 태스크를 보증할 수 없을 때 호출된다. 태스크 제거 모듈은 수행 대기 큐에서 디스페취 되기를 기다리는 비주기 태스크를 제거하여 surplus를 증가시켜 도착한 태스크를 보증하려 하는 것으로 다음의 3가지 정책을 사용할 수 있다.

- HCF(Highest Criticalness First) 정책

도착한 태스크보다 긴급도가 낮은 태스크의 제거

• LSF(Latest Start-time First) 정책

도착한 태스크보다 Latest Start-time이 늦은 태스크의 제거

• EDF(Earliest Deadline First) 정책

도착한 태스크보다 데드라인 시간이 늦은 태스크의 제거

본 논문에서는 EDF 정책을 사용한다.

4. 구현 및 분석

제안 시스템의 구현을 위하여 서버 시스템으로 Micro Soft Windows NT 기반 IIS 4.0 웹 서버와 MS-SQL 6.5를 데이터베이스로 사용하였으며, Windows 98 기반의 클라이언트 시스템을 구현하였다. 데이터 흐름 제어 스크립트는 ASP 프로그램으로 웹 서버에 등록하고, V C++ 언어로 메일 브라우저와 프로세스 엔진을 구축하였으며, 시뮬레이션에서의 함수는 SMPL을 이용하였다.

그림 14는 메일 브라우저를 나타낸 것으로서 GUI 방식을 통하여 데이터 흐름제어 스케줄 정보를 작성하는 과정을 보인다.
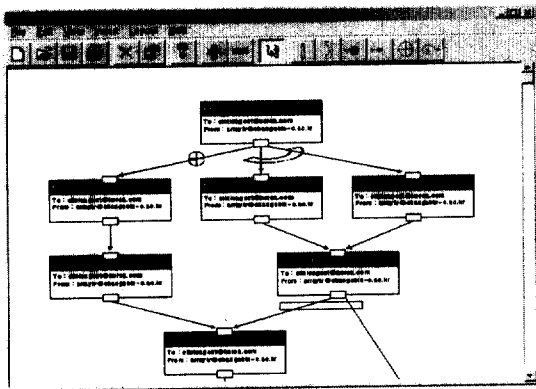
그림 14. 프로세스 흐름제어 스케줄 작성

메뉴와 작업 도구상자에 있는 아이콘으로 프로세스를 디자인하고, 태스크 단위로 의미를 부여한다. 각 태스크 정보는 정의에 따라 설정되며 수행 노드에서 수정 가능하고, 태스크 정보 테이블에서 정의된 내용을 포함하여 스크립트 언어를 생성시킨다. 그 생성된 결과는 그림 15와 같다.

```
begin
[start t0] to [t1 done none]
[start t1] to [t2 mkmsg global]
[start t1] to [t3 mkmsg] and [t4 mkmsg]
[start t2] to [t5 mkmsg]
        ......
end

t1:
{
start: 20010726 12:10
deadline: 20010726 17:00
remainder: 20010726 17:10
 ......
} ......

t1:
{
from: hlee@koje.ac.kr
to: arrayiv@changshin-c.ac.kr
subject: =?EUC-KR?B?M7/5wNrBpiCxuMDUILDoyLk=?=
cc: hlee@koje.ac.kr
MIME-Version: 1.0
Content-Type: text/plain;
Content-Transfer-Encoding: quoted-pritable
        ........
} .......
```

그림 15. 프로세스 스크립트

스크립트 언어는 크게 작업 흐름 모듈과 스케줄 모듈 및 메시지 모듈로 나누어진다. 작업 흐름 모듈은 프로세스의 태스크 정보를 나타내고, 스케줄 모듈은 태스크의 속성 정보를 나타내며, 전송될 메시지는 메시지 모듈에서 헤더와 바디 부분으로 구성된다.

시뮬레이션 모델의 파라미터는 전체 웹 서버노드 3개가 자체 프로세스 엔진을 지니고 있고, 프로세스 발생이 동적으로 생성되며, 내부 태스크는 주기적 태스크의 주기 값을 각 노드마다 다르게 설정하였다. 비주기적 태스크의 평균주기는 그 값을 평균값으로 하여 지수분포로 태스크가 도착하도록 하였으며, 또한 서브 태스크의 클러스터링 과정과 전체 알고리즘 수행을 위한 오버헤드를 고려하였다. 결함 발생 서비스는 전체 시뮬레이션 시간의 67%(임의적) 경과시 한 곳에서 발생하는 것으로 하였다. 서브 태스크 작성을 위해 선형 클러스터링 알고리즘을 이용하였으며[19], 프로세스 발생은 과부하 상태가 되도록 하였고, 설계 알고리즘은 태스크 수행 보증율로 평가하였다.

그림 16은 수행 보증율을 나타내며, 전처리기를 지니고 있는 제안 시스템(Used F.T)은 이를 적용하지 않은 경우에 비하여 보증율이 향상되었다.
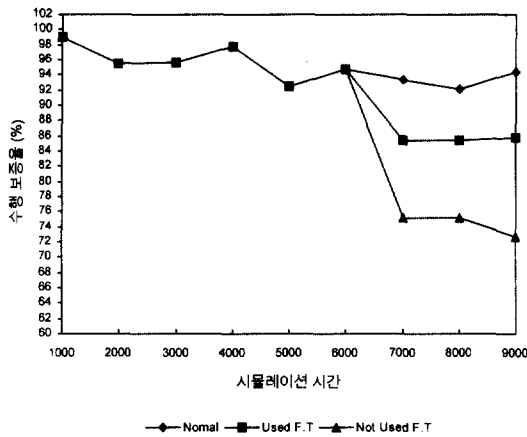
그림 16. 결함 발생 시 태스크 수행 보증율

그림 17은 정상상태에서 태스크 제거 모듈 및 전처리기(S_T_M)의 사용 여부에 따른 수행 보증율을 비교한 것으로서, 시뮬레이션 시간에 따라 유동적임을 알 수 있다. 이는 태스크 제거 모듈 및 전처리기가 지니고 있는 오버헤드에 따른 처리능력 감소에 의한 것으로서 수행 보증율의 측면에서는 양호한 결과가 아니지만, 실시간 시스템에서의 우선 순위가 높거나 데드라인 만족이 필수적인 경우에 유용하다.
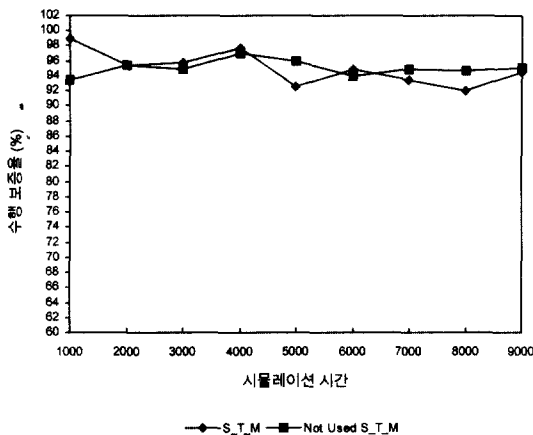
그림 17. 정상상태에서의 수행 보증율

5. 결 론

본 논문에서는 웹 서버 환경과 분산시스템에 적합한 분산워크플로우 시스템을 제안하였다. 제안 시스템은 수행 보증율을 향상시킬 수 있는 프로세스 엔진과, 프로세스 디자인을 위한 그래픽 아이콘을 지닌 메일 브라우저로 구성되어 있으며, 메일 시스템 기반의 스크립트 생성기를 지니므로 신속한 프로세스의 수행이 가능하다.

또한, 메일시스템과 프로세스의 디자인을 관련 업무 조직도와 조합하여 활용 가능하며, 전처리기를 통한 시스템의 수행 결합이 허용된다. 서비스가 사용자에 의해 수행되어야 할 경우 대리인이 업무 수행을 대행할 수 있도록 하여 분산워크플로우 시스템의 신뢰성과 가용성을 향상시키며, 정형 및 비정형 데이터를 업무프로세스에 따라 수집 · 분배 및 라우팅할 수 있는 기능을 가지므로 업무 흐름의 자동화가 가능하다.

향후, 프로세스의 각 단계마다 다양하게 존재하는 규칙과 조건을 나타낼 수 있는 프로세스 디자인 도구와 프로세스 제어 정보를 위한 데이터베이스 및 수행 오버헤드를 감소시키는 연구가 필요하다.

참 고 문 헌

[1] R. Friedman and D. Mosse, "Load Balancing Schemes for High-Throughput Distributed Fault-Tolerant Servers," Journal of Parallel and Distributed Computing, pp. 475-488, Dec. 1999.

[2] V. Cardellini, M. Colajanni and P.S. Yu, "Dynamic Load Balancing on Web-server Systems," IEEE Internet Computing, pp. 28-39, May 1999.

[3] R. Buyya, "High Performance Cluster Computing: Architectures and Systems," Prentice-Hall, p. 849, 1999.

[4] L. Fischer, The Workflow Paradigm *The Impact of Information Technolory on Business Process Reengineering, Future Strategies*, Inc., Alameda, CA, 2nd. edition, 1995.

[5] C. Mohan, Tutorial, *State of Art in Workflow Management System Research and Products*, IBM Almaden Research Center, Workflow Systems and Interoperability, Istanbul, Turkey, August 1997.

[6] David Hollingsworth, *Workflow Management*

Coalition Specification : The Workflow Reference Model, WfMC, 29 Nov. 1994.

[7] Nortel, *Workflow Management Facility Specification*, OMG, 1997.

[8] M. Zisman, *Representation, Specification, and Automation of Office Process*, Ph.D.dissertation, Wharton School, Univ. Pennsylvania, 1997.

[9] C. Ellis, M. Bernal, "Officetalk-D : An Experimental office information system, Proceeding of the ACM-SIGOA Conference on Office Information System", pp. 281-288, Nov. 1992.

[10] Jintae Lee, Michael Gruninger, Yan Jin, Thomas Malone, Austin Tate, Gregg Yost, The PIF Process Interchange Format and Framework v 1.1, PIF working Group, May 24, 1996.

[11] Graig Schlenoff, Amy Juntilla, Steven Ray, *Unified Process Specification Language : Requirements for Modeling Process*, NIST, 1996.

[12] 김기봉, IRDS 기반 정보 저장소와 CASE 도구 통합에의 활용, 박사학위 논문, 충남대학교, 1998.

[13] Mark R. Jones, "Unveiling Repository Technology", Database Programming and Design, pp. 28-35, April 1992.

[14] P. K. Lala, *Fault Tolerant Fault Testable Hardware Design,* Prentice-Hall.

[15] Leinbaugh D.W, M.R. Yamani, Guaranteed response in a Distributed Hard Real-Time Environment, Proc. IEEE Real-Time System Symposium, Dec. 1982.

[16] D. Andresen, T. Yang, V. Holmedahl and O. Ibarra, "SWEB: Towards a Scalable WWW Server on Multicomputers," Proceedings of ISPP, IEEE, pp. 850-856, April 1996.

[17] B. Narendran, S. Rangarajan and S. Yajnik, "Data Distribution Algorithms for Load Balanced Fault-Tolerant Web Server," Proceedings of the 16th Symposium on SRDS, Oct. 1997.

[18] 박규석, 김성후, "Web에서 데이터 흐름제어가 가능한 Mail Browser의 설계 및 구현", 정보처리학회논문지, 제6권, 제10호, pp. 2752-2763, 1999.

[19] Kermal Efe, Heuristic Models of Task Assignment Scheduling in Distributed Systems, Computer, June 1982.

이 헌

1987년 경남대학교 공과대학 전자계산과(학사)
1992년 경남대학교 컴퓨터공학과(석사)
1997년 경남대학교 컴퓨터공학과(박사수료)
1993년~현재 : 거제대학 전자계산과 부교수, 산학처장
관심분야 : 멀티미디어시스템, 분산시스템, 워크플로우시스템

박 규 석

1980년 중앙대학교 전자계산학과 전산학전공(석사)
1988년 중앙대학교 전자계산학과 전산학전공(박사)
1982년~현재 경남대학교 정보통신공학부 교수
1992년~1996년 경남대학교 전산정보원 원장
1995년~1996년 한국정보과학회 이사, 영남지부장
1999년~현재 경남대학교 정보통신연구소 소장
관심분야 : 분산처리시스템, 정보통신 소프트웨어, 멀티미디어 시스템